#19

Appeal 10/13/03

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of

Phillip MERRICK, et al. Attorney Docket A7145

Appln. No.: 09/274,979

Confirmation No.: 9188 Group Art Unit: 2151

Filed: March 23, 1999 Examiner: COURTENAY

For: XML REMOTE PROCEDURE CALL (XML-RPC)

**DECLARATION**

I, Joseph Lapp, hereby declare that:

1. I am a named inventor of the U.S. Patent application identified above ("the present application").

2. I have read and understood the reference referred to by the examiner in the Office action of February 27, 2003 as "Goldfarb et al," which is an excerpt from a publication entitled "The XML Handbook."

3. I did write this portion of the Goldfarb et al publication, and provided all substantive content of this portion of the publication. I did so at a time after the invention claimed in the present application was made, so that the substantive content of this publication excerpt reflects contributions from all of the co-inventors of the present application.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

3/19/2003
Date

Joseph Lapp


# The Forthcoming Metadata Revolution

**ABSTRACT** – Today's middleware is designed to bring interoperability to existing applications in a heterogeneous environment. While necessary, it will eventually succumb to the needs of living systems that require a more dynamic medium for data interchange. The formation of this new medium can start to be seen in the Internet using open protocols. Here the two most important technologies for the next generation middleware exist – Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML).

## *Middleware Trends*

Middleware is a widely used term to define a set of software that facilitates communication between two distinct applications that do not natively communicate. Use of middleware enables applications to be built in a modular manner, thus creating highly reusable software components. In addition, middleware sometimes simplifies software development by handling extremely complex details such as transactions, queuing, or storage. Regardless, this growing class of software has become very important to network computing today.

When we discuss middleware today, we are primarily discussing the category that includes the following: database/SQL, asynchronous messaging, publish and subscribe, object request brokers, and transaction processing monitors. These have evolved from the need to tie together disparate applications in a heterogeneous computing environment. Extending from this evolution, we have identified two critical trends; the consolidation of known middleware into a single entity and the emergence of more sophisticated data-centric middleware.

The consolidation of middleware classifications is an inevitable evolution, since middleware vendors cannot survive selling merely one middleware type. Moreover, their base of satisfied customers are requiring them to provide additional functionality for their existing investment. For example, Oracle has grown outside of its database origins to offer transaction processing, asynchronous messaging, publish and subscribe, and an ORB– all with the database being the central facilitator of these technologies. Consequently, object request broker vendors are adding these middleware types as services for their ORB.

The second trend, the move to data-centric, stems from the fact that the previously mentioned middleware classes are primarily service based; that is, they exist as a set of network services with the focus on procedural tasks. This can be likened to early stages of programming with structural programming languages such as COBOL and PL/1. With the focus becoming data-centric, the emphasis is on the content itself, that is living systems with the procedural entities being defined by the data itself. This can be likened to the shift to object-oriented programming, where objects represent data and the procedures represent behavior of object.

## *What Is A Living System?*

Living systems are just emerging as a science, but as with many other aspects of computing, are based on a large field of scientific research. For years the scientific community has looked at individual sub-sections of systems, for example, biology examines cells in relations to their containers. Living systems research states that nothing exists in a vacuum, and when examined this way, do not yield their actual behavior as when examined interacting inside their life system.

NC.Focus is a technology research and IT Management consulting firm specializing in assisting companies to maximize their investment in Network Computing. NC.Focus offers subscription-based research and on-site consulting that provides innovative concepts for more effective computing strategies. Additional information and subscriptions can be obtained via the NC.Focus Web site (http://www.ncfocus.com) or by calling 516-792-0997.

Most of our existing computer applications were designed to solve an immediate problem. This problem may have been enterprise-level or simply departmental in scale. Regardless, the resulting systems were very static and required a large complement of middleware to join together. In contrast, a computer application modeled after a living system is capable of growing and evolving without requiring extreme re-designs or re-architecture.

Building a living information system requires the use of metadata, that is, data that describes itself. This requirement is best shown by examining an attempt to learn about a system that uses a proprietary data format. For example, if a user receives a word processing file generated by Microsoft Word, it is highly unlikely that the user will be able to use that data without first having Word, a Word translator, or a Word viewer. However, the same document provided in metadata format, such as HTML, can be explored and used, even without an HTML browser, that latter just provides added value.

When data is self-describing, it lends itself to be readily incorporated into new system components. These components can blend in and interact with other components through the metadata, without having intimate knowledge of each other. This is a benefit of object-oriented design in general, it is just most efficiently executed when metadata is used to represent the object.

### *The Shift To Data-Centric Computing*

Middleware (as it is defined today) represents a class of software that must eventually face extinction; but this is a much longer-term trend. Since the existing class of middleware was designed to bring interoperability to systems that did not plan for communications during its design phase, it is merely a Band-Aid that will remain in effect until these systems are decommissioned. Future systems are already being designed with the full knowledge that they will co-exist with all the other systems in the enterprise, and perhaps even beyond that boundary.

Learning from the mistakes of the past, these newly designed systems are more heavily concentrated on reuse of data than reuse of code. A single accounting record may be utilized in the inventory system, the enterprise resource planning (ERP) system, and in the sales support system. This demonstrates a need for a central repository that is shared among all these systems. As companies go through re-engineering efforts, their legacy systems are slowly being replaced by systems that understand this need. It is important to note here that this repository may have no coherence that it is holding an accounting record. Indeed, it may be holding many different data types. It will be up to each individual system to utilize the records in the repository as needed. Furthermore, in contrast to traditional systems, the accounting system does not own the accounting record; it is merely the input to the repository.

The technologies that will satisfy this requirement, and eventually become the next leading middleware solution, are based upon the Standard Generalized Markup Language (SGML), the forefather of Hypertext Markup Language (HTML). SGML is also the forefather of a newer grammar known as Extensible Markup Language (XML), which is a key technology to building the types of repositories discussed above. XML uses SGML grammar to build sequences of name/value pairs. These name/value pairs can be created in a hierarchical manner allowing for scoping of data sets. For example, the accounting record mentioned previously might have four different name/value pairs, such as payer, date, amount, and account. The scope of these name/value pairs only makes sense within the context of an accounting record; hence, these requirements form a schema that needs to be adhered to.

The following is an example of how XML can be used to represent an accounting transaction.

XML Example

```
<Transaction>
    <Account_Number="012120"/>
    <Account_Title="J.P. Morgenthal"/>
    <Payer_Title="Joe's Garage"/>
    <Payer_Account="102432"/>
    <Payer_ABA="0164450232"/>
    <Date="11/01/97"/>
    <Amount="100.00"/>
</Transaction
```

The previous example can be used in its shown form, known as a "well-formed" XML document, or combined with a formal description of the Transaction schema called a DTD (Document Type Definition). When used with its DTD, an XML document is known as "valid". More importantly, together, all these attributes create a self-describing piece of data that can be evaluated in a number of different ways. Moreover, systems do not need to have a preconceived notion of the data's structure--only the name/value pairs it is interested in and a DTD. This illustrates the power of XML; generic viewers can be built to view new data types simply by publishing a DTD.

As with all technology changes, there will always be skeptics and biases. But for those who may not believe that this trend is possible or even real, all one has to do is surf the Web. The World Wide Web has become the most heavily used repository in the history of computing. The data available on a Web page is used by people for viewing, search engines for cataloging, and by Java as a container. Each application views the Web data differently.

### *Who Will Lead This Transition?*

As with all issues regarding the Web, the W3C will be in charge of defining XML standards, as well as adopting certain DTDs. However, the largest corporation standing behind XML today is Microsoft. Already, Microsoft has pushed to standardize two important DTDs: Channel Data Format (CDF), which Microsoft uses to deliver data to their Active Desktop environment along with Internet Explorer 4.0 (and which will be a standard component of Windows 98--barring intervention by the Department of Justice), and the Open Software Description (OSD) format, which Microsoft has partnered with Marimba to define and deliver. OSD is a format used to make updating software easier to accomplish over the network.

Netscape Communications Corp. has also pledged some degree of support for XML as well. They have delivered to the W3C a working draft of a proposal for its Meta Content Framework (MCF), which outlines using XML for building relationships between Web-based data. This specification has been incorporated into a larger one called the Resource Description Format (RDF), for which there is now a W3C working group to move it into standard.

In addition, there are smaller factions that are attempting to use XML for everything from Electronic Data Interchange (EDI), which is a very expensive and difficult system to build and maintain, to distributed object computing.

### *Where's The Middleware?*

Metadata middleware will be defined by four key components: translators, filters, transporters, and repositories. Translators are code that take as input XML and DTDs and convert them to other usable forms. Filters will take data from a native store and publish them in XML form. Transporters are responsible for moving the data

based upon content, for example, workflow engines, e-mail routing, and wide-scale Web distribution. And repositories are the storage facilities that will allow access to and storage of metadata.

As for companies delivering products based upon this technology:

Microsoft's Internet Explorer 4.0 and Active Desktop environment are based on the delivery of content to the desktop in the form of XML. Microsoft has made a strong commitment to XML. Indeed, their entire Web strategy for platform-independent application development is to use XML, Dynamic HTML, and scripting. In addition, Microsoft has been very active in submitting proposals to the W3C to enhance XML's benefits.

DataChannel is a Seattle, WA-based company currently shipping ChannelManager, which is an XML-based administration tool for publish and subscribe systems. They are also working on delivering software that will publish existing databases in XML, utilizing the database's table definitions to dynamically define the DTD. In addition, the company is examining use of XML to build distributed object applications.

webMethods is a Fairfax, VA-based company that sells a tool, called the *Web Automation Toolkit*, for developing distributed object applications using open Web protocols. WIDL (Web Interface Definition Language) is an XML-based language that is used to generate application components that pass data between applications using HTTP as its transport layer. webMethods has submitted its specification for WIDL to the W3C for recommendation as a standard (product review follows).

Of note, this trend is not born of the Web. Indeed, metadata has been an integral part of directory services for years. Standards like X.500 utilize name/value pairs to describe their objects and facilitate searches. Clearly, this move toward data-centric computing will simultaneously raise awareness of the requirement and utility of this service. Eventually, the directory service and the metadata repository will become one. Watch both Microsoft and Novell as key drivers on this direction.

### *Dueling Metadata Directions*

This research note focuses heavily on the use of XML to satisfy metadata requirements. From our research, this language is very powerful and offers the capability to create both metadata descriptions and is itself a meta-language. Previously, these capabilities have only been seen in heavy-duty object-oriented environments like Smalltalk. However, there is another group that calls itself the Metadata Coalition that is attempting to define a language for interchange of data stored in data warehouses.

NC.Focus queried the Metadata Coalition to find out if they had plans to adopt XML to accomplish their mission. The Metadata Coalition is now on revision 1.1 of its specification, and does not have plans to move to XML in the future. Normally, this would not present a problem; however, one of the supporters of the Metadata Coalition is Microsoft Corp, the same group promoting XML.

It turns out that two different groups within Microsoft are both working on metadata issues, the Platform group and the Tools group. Microsoft's involvement with the Metadata Coalition is centered on Microsoft's Repository, which ships with their Visual Basic 5.0 product, which is administered by the Tools group. This repository has a design for storing meta-information about components that are stored within it, which has been adopted as a standard by the Metadata Coalition. XML efforts are being administered by the Platform group.

Microsoft is currently examining the intersection of these two technological directions internally, and will address it in the near future.

***Resources For Tracking The Metadata Revolution***

**DataChannel - xml.datachannel.com**
**Microsoft - www.microsoft.com/standards/xml**
**Robin Cover's XML Site - www.sil.org/sgml/xml.html**
**webMethods - www.webmethods.com**
**W3C - www.w3c.com. The World Wide Web consortium, led by Web creator Tim-Berners Lee, this group is charged with adopting the standards that will drive the Web and its applications.**
**The XML/EDI Group - www.geocities.com/WallStreet/Floor/5815. A small group dedicated to promoting XML/EDI tools. Their mailing list is available for subscription via their Web site and is contributed to by some of the leading drivers of XML**

## *webMethod's Web Automation Toolkit*

Distributed object computing (DOC) used to belong to the programming elite. Requirements for understanding programming languages like C, C++, and Smalltalk, along with network protocols and messaging, made this technology elusive except to a small number of well-trained computer scientists. webMethods is one of the drivers behind a revolution of products based upon open Internet protocols that will allow a much larger base of developers to build distributed object applications.

The Web Automation Toolkit is a unique tool that falls into the category of code generators, but that is just one of it powerful capabilities. At the core of The Toolkit is webMethod's specification for defining Web-based interfaces called WIDL (Web Interface Definition Language). The name is derived from its closely akin cousin, the Common Object Request Broker Architecture (CORBA) IDL, which is used to define the inputs and outputs of a method (function) on a remote object. Passing data as parameters to methods is the way common distributed object applications work today. WIDL is used to define inputs and outputs of Web-based applications.

Understanding The Toolkit's capabilities requires examination of development of two different types of applications. First, a client application that utilizes information from the Web in an automated manner, and second, a distributed application that utilizes Web protocols for formatting and passing data.

## *Using The Toolkit To Process Web-based Data*

The Web is a goldmine of static and dynamic data that takes unlimited hours to extract small morsels of useful information. Being able to automate this extraction is a valuable commodity – one supplied by webMethod's Web Automation Toolkit. The toolkit is 100% Java (currently under testing for the 100% Pure Java logo) application that reads in multiple document types (such as text, HTML, and XML) and maps them into webMethod's document object model (DOM). A DOM assigns various parts of a Web page to variables that can be used for programmatic access. For example, webMethod's DOM provides a way to reference the page title, cells in a table, and form fields. webMethod's is working closely with the W3C on the definition of a standard document object model for the Web, which is currently a working group within the W3C.

The DOM mapping is represented using webMethod's XML-based WIDL. This is an ASCII description that defines the inputs to a Web application and maps the returned data set to internal variables. Obviously, mapping data only to absolute locations on the Web page would provide limited utility in face of the Web's constant cosmetic makeovers. WIDL offers a number of methods for extracting data including ranges (based upon row and column positions), regular expression masks (locate a string that matches a pattern), and a straightforward text stream of the entire page.

WIDL also allows for inclusion of certain logical operations over the returned data. For example, a timeout or "server busy" message might require a retry. Conditions allow determination of success or failure of an attempted retrieval as well as defining an action for each.

Once the WIDL is defined for a particular Web page, or site, it can be used to generate executable code. webMethods generates bindings for four different programming languages: C++, Visual Basic, Javascript (with Java), or Java. The Toolkit also supports output for use with Microsoft Excel and Sybase Powerbuilder. This code handles accessing the remote site, supplying the input, and mapping the output to a set of variables inside of the application. *Basically, this eliminates most of the tedious programming work associated with building distributed applications.*

The Java/Javascript capability was tested by NC.Focus' Labs to automate retrieving electronic mail from a POP3 server using our CGI-based POP3 reader. Total time to build the application was one hour.

## *Building Distributed Object Applications With The Toolkit*

Using XML, HTML, and HTTP for building distributed applications is quickly gaining support by the development community. Instead of defining rigid C++ applications that simply make remote method calls, these protocols allow for the creation of dynamic objects. A dynamic object is one that can change during its lifetime, thus creating new attributes. If data are viewed as reusable entities, it is very expensive to first have to copy a piece of data to enhance it. XML and HTML both offer grammars that can define an encapsulated unit of data, but can also add and subtract data without "breaking" their receivers.

This capability can be extended directly into applications bypassing the Web server and the Web browser. That is, Hypertext Transfer Protocol (HTTP) can be used to "POST" and "GET" data from a remote application. Consequently, the remote application can utilize XML and HTML to describe the passed data.

WIDL and the Web Automation Toolkit generate the interface and code for passing data between two applications using HTTP. Primarily this will be of interest to organizations that have been "Web-izing" their existing base of applications. And most likely these applications will utilize a Web server as a medium to transfer the data to the application via one of the popular communications gateway protocols (NSAPI, ISAPI, server-side Java, or CGI). Still, there are benefits to designing applications to work in this way:

The application is extremely dynamic. Any data available inside of a HTTP stream is available for incorporation into a client application.

The receiving component of the application is very modular. It can work as a CGI process or embed its own HTTP receiver and use The Toolkit generated code to parse the HTML/XML into values.

HTTP provides an access mechanism to data stored in a diversity of data formats. Data stores that provide an HTTP interface all allow data retrieval in a consistent manner. This greatly reduces efforts across a large number of platforms.

## *webMethods Future Direction*

Like NC.Focus, webMethods agrees that server-side Java is a much richer and stronger environment for application development. Relative to this fact, webMethods is ensuring that future versions, starting with the soon to be released version 2.1, will ease development of server applications as well as client applications.

The concept of the Weblet is primarily used as a client for accessing a Web Automation Server and retrieving data using a WIDL mapping. Future versions of the Web Automation Toolkit will support development of Servlets (not

to be confused with the Sun specification for web-server processes), which will allow for both automated retrieval of data, but also automated updating of that data as it changes. Solid examples of this functionality can be found on webMethod's web site (http://www.webmethods.com/demos/News/index.html for one).

*Conclusions*

+ Web Automation Toolkit is an excellent method of developing distributed applications that use HTTP as a transport and HTML or XML as a data format. In addition, The Toolkit facilitates extracting data from the Web in a logical manner without an expensive development effort.

- Many existing corporate applications have not been converted for use with the Internet/Intranet. Processing cannot be embedded completely into the client application, thus requires a server process running either locally or remote.